AF 2188
IFW

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicant: DOV MORAN §

Serial No.: 09/922,153 §

Filed: August 6, 2001 § Group Art Unit: 2188

For: NOVEL FLASH MEMORY ARRANGEMENT § Attorney Docket: 246/158

Examiner: Pierre M. Vital §

TRANSMITTAL OF APPEAL BRIEF

Commissioner of Patents and Trademarks
Washington, DC 20231

Dear Sir:

Transmitted herewith in triplicate is the APPEAL BRIEF in this application with respect to the Notice of Appeal filed on June 18, 2004.

The application is on behalf of

_X_ other than a small entity

___ small entity

verified statement:

___ attached

___ already filed

Pursuant to 37 CFR 1.17(f) the fee for filing the Appeal Brief is:

___ small entity $ 165

_X_ other than a small entity $ 330

Appeal Brief fee due $ 330

09/08/2004 SDENBOB1 00000123 062140 09922153
02 FC:1251 110.00 DA

_X_ Applicant petitions for an extension of time under 37 CFR 1.136 for the total number of months checked below:

	small entity	not small entity
_X_one month	$ 55	$ 110
___two months	$ 210	$ 420
___three months	$ 475	$ 950
___four months	$ 740	$ 1480

If an additional extension of time is required please consider this a petition therefor.

The total fee due is:

Appeal brief	$330
Extension fee (if any)	$110
TOTAL FEE DUE	$440

Please charge Account No. 06-2140 the sum of $440. A duplicate copy of this transmittal letter is attached.

If any additional extension and/or fee is required, this is a request therefor and to charge Account No. 06-2140.

If any additional fee for claims is required, please charge Account No. 06-2140.

Respectfully submitted,

Mark M. Friedman
Attorney for Applicant
Registration No. 33,883

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicant:	§	
	§	
DOV MORAN	§	
	§	
Serial No.: 09/922,153	§	
	§	
Filed: August 6, 2001	§	Group Art Unit: 2188
	§	
For: NOVEL FLASH MEMORY ARRANGEMENT	§	Attorney Docket: 246/158
	§	
Examiner: Pierre M. Vital	§	

Commissioner of Patents and Trademarks
Washington, DC 20231
ATTENTION: Board of Patent Appeals and Interferences

APPELLANT'S BRIEF

Dear Sir:

This is in furtherance of the Notice of Appeal filed in this case on June 18, 2004.

The fees required under § 1.17(f) and any required petition for extension of time for filing this brief and fees therefor are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief is transmitted in triplicate.

This brief contains these items under the following headings and in the order set forth below:

I. REAL PARTY IN INTEREST

II. RELATED APPEALS AND INTERFERENCES

III. STATUS OF CLAIMS

IV. STATUS OF AMENDMENTS

V. SUMMARY OF INVENTION

VI. ISSUES

VII. GROUPING OF CLAIMS

VIII. ARGUMENTS

_X_ ARGUMENT: VIIIA REJECTIONS UNDER 35 U.S.C. 102

_X_ ARGUMENT: VIIIB REJECTIONS UNDER 35 U.S.C. 103

IX. APPENDIX OF CLAIMS INVOLVED IN THE APPEAL

I. REAL PARTY IN INTEREST

The real party in interest in this case is:

M-Systems Flash Disk Pioneers, Ltd.

Central Park 2000

Atir Yeda 7

44425 Kfar Saba

Israel

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals and interferences.

III. STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

There are 36 claims in the application.

B. STATUS OF ALL THE CLAIMS

1. Claims cancelled: 2, 31-35.

2. Claims withdrawn from consideration but not cancelled: none

3. Claims pending: 1, 3-30, 36

4. Claims allowed: none

5. Claims rejected: 1,3-30, 36

C. CLAIMS ON APPEAL

The claims on appeal are: claims 19, 22-28 and 36.

IV. STATUS OF AMENDMENTS

In response to the Official Action mailed on January 28, 2004, claims 31-35 were canceled and new claim 36 was added.

V. SUMMARY OF INVENTION

The present invention is a flash memory device for storing boot code to be executed by a processor and a method of booting a system that includes the flash memory device and the processor. The device includes a flash memory, for storing the code, such that the code cannot be executed in place from the flash memory. The device also includes a volatile memory to which the code is copied for execution, a logic, separate from the processor, for doing the copying, and a bus via which the flash memory, the volatile memory and the logic communicate directly with each other. When the system is booted, a busy signal is sent to the processor and is not removed until enough code to start booting has been copied to the volatile memory.

VI. ISSUES

The issues presented for review are:

1. Whether claims 22 and 23 are anticipated by Garfunkel et al., US Patent No. 6,615,404 (henceforth, "Garfunkel et al. '404")

2. Whether claims 19 and 36 are unpatentable over Garfunkel et al. '4040 and Chieng et al., US Patent No. 6,035,346 (henceforth, "Chieng et al. '346")

3. Whether claims 24-28 are unpatentable over Garfunkel et al. '404 and Lee, US Patent No. 6,370,645 (henceforth, "Lee '645")

VII. GROUPING OF CLAIMS

For the purpose of both the § 102(b) rejections and the § 103(a) rejections, all claims are grouped separately, and stand or fall on their own merits, except that if a claim falls, then all the claims that depend from that claim also fall.

VIII. ARGUMENTS

REJECTIONS UNDER 35 U.S.C. 102

The Examiner rejected claims 22 and 23 under § 102(b) as being anticipated by Garfunkel et al. '404.

Garfunkel et al. '404 teach an embedded system **10** and a method of upgrading the software of the system. Embedded system **10** includes a controller **11**, a flash memory **12** and a RAM **13**. The current boot code of embedded system **10** is stored in an ordinary sector **21** of flash memory **12**. The original boot code of embedded system **10** is stored in a special sector **20** of flash memory **12** that is hardware-protected, for example by requiring an unusually high voltage for reprogramming. If an updating of the system software in flash memory **12** is unexpectedly interrupted, the original boot code is available as a backup in sector **20** to re-start the system.

The primary difference between the teachings of Garfunkel et al. '404 and the present invention, as recited in independent claim 22, is that claim 22 requires the flash memory to be such that code can not be directly executed therefrom. It is clear from Garfunkel et al. '404 that flash memory **12** is executable. See, for example, column 6 line 66 through column 7 line 3:

> According to another preferred embodiment of the invention, one or more sectors **22** of the flash memory **12** are also copied into the RAM **13**. In this case, the controller **11** operates that section of the program from the RAM **13**, rather than from the flash memory **12**. (emphasis added)

In other words, during normal operation of embedded system **10**, as opposed to when the code in sectors **22** of flash memory **12** is being upgraded, controller **11** has the option of executing the code directly from those sectors **22**. The Examiner misinterpreted the subsequent lines (column 7 lines 5-8) in Garfunkel et al. '404,

> In current practice, during updating, software can not be run from the flash memory **12** at all, and therefore, only software copied into RAM **13** is functional.

as indicating that flash memory **12** is not directly executable. The operative word in this passage is "during updating". The point of this passage is that in order to update the code in flash memory **12**, the old code must be erased, and so is not available for execution. During normal operation of the system, as opposed to during an update of the code in flash memory **12**, the code in flash memory **12** is directly executable. Thus, the present invention, as recited in independent claim 22, is not anticipated by Garfunkel et al. '404.

Furthermore, the present invention, as recited in independent claim 22, is not even obvious from Garfunkel et al. '404. It is not obvious from Garfunkel et al. '404 that a non-executable flash memory can be used to store boot code. As best understood, during normal operation (as opposed to immediately following an interrupted software upgrade), the boot code of Garfunkel et al. '404 must be executed directly from sector **21** of flash memory **12** because RAM **13**, being volatile, is blank when the system is powered up. As stated in column 4 lines 44-46,

> The flash memory **12** is the only non-volatile memory which is required for storing the operating software and/or other desired instructions. (emphasis added)

In other words, no other non-volatile memory need be available for executing boot code during normal system power-up. Therefore, flash memory **12** must be executable.

The present invention gets around this problem by including logic **42** to copy boot code from non-executable flash memory **14** to executable SRAM **40** when system **30** of the present invention powers up. There is neither a hint nor a suggestion of special circuitry analogous to logic **42**, separate from controller **11**, in embedded

system **10** of Garfunkel et al. '404 for copying boot code from sector **21** of flash memory **12** to RAM **13** on power-up.

In response to these arguments, the Examiner cited column 7 lines 24-38 of Garfunkel et al. '404 as teaching that the boot code normally is executed from RAM **13**. This passage from Garfunkel et al. '404 cannot mean that <u>all</u> the boot code is executed from RAM **13**. As noted above, RAM **13**, being volatile, is blank when the system is powered up. Therefore, at least the initial portion of the boot code, *i.e.*, enough of the boot code to enable controller **11** to copy the rest of the boot code to RAM **13**, must be executed from flash memory **12**.

With independent claim 22 allowable in its present form, it follows that claim 23, that depends therefrom, also is allowable.

REJECTIONS UNDER 35 U.S.C. 103

The Examiner rejected claim 19 under § 103(a) as being unpatentable over Garfunkel et al. '404 and Chieng et al. '346.

As noted above, flash memory **12** of Garfunkel et al. '404 is executable. Therefore, claim 19, that recites a non-executable flash memory, is not obvious from Garfunkel et al. '404 either alone or in combination with any other reference.

Even if Garfunkel et al. '404 had taught a nonexecutable flash memory for storing boot code, claim 19 would not be obvious from the combination of Garfunkel et al. '404 and Chieng et al. '346. Chieng et al. '346 teach a computer system **500** that includes a PCI device **505** whose PROM **520** can be reprogrammed by a host processor **105**. For this purpose, host processor **105** sends a busy signal to PCI device **505** to put the CPU **510** of PCI device **505** into a wait and hold state, downloads reprogramming code to the RAM **515** of PCI device **505**, initializes a memory signature in RAM **515**, and releases CPU **510** from the wait and hold state. This sequence of signals and code flow is opposite to the sequence recited in claim 19. Claim 19 recites sending a busy signal to a processor, transferring boot code from a flash memory to a volatile memory, removing the busy signal, and executing the boot code by the processor. In the context of Garfunkel et al. '404, this would be sending a busy signal to controller **11**, transferring boot code from flash memory **12** to RAM **13**, removing the busy signal, and executing the boot code by controller **11**. But the analog, in computer system **500** of Chieng et al. '346, of controller **11** of Garfunkel et al. '404, is not CPU **510** but rather host processor **105**. In computer system **500** of Chieng et al. '346, host processor **105** is the source of the busy signal, not the target of the busy signal. One ordinarily skilled in the art would not be led by a study of

Chieng et al. '346 to contemplate sending a busy signal to controller **11** of Garfunkel et al. '404 to allow transferring code to a memory for execution by controller **11**.

With independent claim 19 allowable in its present form, it follows that claim 36, that depends therefrom, also is allowable.

The Examiner rejected claims 24-28 under § 103(a) as being unpatentable over Garfunkel et al. '404 and Lee '645.

As noted above, flash memory **12** of Garfunkel et al. '404 is executable. Therefore, claims 24-28, that recite a non-executable flash memory, are not obvious from Garfunkel et al. '404 either alone or in combination with any other reference.

Even if Garfunkel et al. '404 had taught a nonexecutable flash memory for storing boot code, claims 24-28 would not be obvious from the combination of Garfunkel et al. '404 and Lee '645. Lee '645 teaches a hard disk drive that includes a microprocessor **10**, a flash ROM **40** and a RAM **42**. As described in column 3 lines 55-60 and in column 4 lines 1-9, flash ROM **40** is just large enough (less than 8 Kbytes) to store boot code for the hard disk drive. As described in column 3 line 50, RAM **42** is considerably larger than this (32 Kbytes). Now, in claims 24-28 it is the volatile memory component, not the flash memory, that is only large enough to store the basic initialization boot code. One ordinarily skilled in the art might be led by a study of Lee '645 to make flash memory **12** of Garfunkel et al. '404 just large enough to store boot code, but not to make RAM **13** of Garfunkel et al. '404 just large enough to store boot code.

In response to these arguments, the Examiner cited Lee '645 column 4 lines 24-29 as teaching that

> ...by using a RAM or volatile memory of a small capacity, the correction and maintenance of the codes are easy and multiple functions are obtained.

The full paragraph that includes these lines is as follows:

> The codes stored in the flash ROM **40** are commands for uploading the codes actually used to operate the HDD, that is, the firmware stored in the maintenance area of the hard disk **18** to the RAM **42**. These codes are accessed by the microprocessor **10**. The codes to be uploaded, namely, the codes stored in the maintenance area of the hard disk **18** are written during a HDD fabricating process. In such a case, the codes are grouped into several segments each having size adequate to be completely uploaded to the RAM **42**. For classification, the codes are divided into a main code and a test code each further divided into several segments. Therefore, the codes can be uploaded at any time, not only during the initial power-on of the HDD, but during the operation of the HDD. During a process test, only the test code is uploaded to the RAM **42**. When the user is using the HDD, only the main code can be uploaded. Then since the codes are divided into structural modules, the correction and maintenance of the codes are easy and multiple functions are obtained even by the RAM of a small capacity. It will be appreciated to those skilled in the art that the RAM **42** may include a dynamic RAM or a static RAM.

It is clear from the full paragraph that the logic of the Examiner's reply is the opposite of what is actually taught by Lee '645. It is not the small size of RAM **42** that allows the code to be uploaded to RAM **42** to be modular, but the modularity of the code to be uploaded to RAM **42** that allows RAM **42** to be small. Note that there are two kinds of code that are uploaded to RAM **42**: boot code from flash ROM **40** and HDD operational code ("the codes actually used to operate the HDD") stored in the maintenance area of hard disk **18**. RAM **42** of Lee '645 needs to be large enough to accommodate either the boot code or the various modules of HDD operational code. Presumably, this is why RAM **42** is four times as large as is needed to accommodate the less than 8 Kbytes of boot code. By contrast, the volatile memory

component recited in claims 24-28 is only large enough to store enough boot code for basic initialization of the system, and no larger.

Respectfully submitted,

Mark M. Friedman
Attorney for Applicant
Registration No. 33,883

Date: September 1, 2004

IX. APPENDIX OF CLAIMS INVOLVED IN THE APPEAL

The text of the claims on appeal is:

19. A method for booting a system, the system featuring a processor for executing boot code, the method comprising:

providing a flash-based unit in the system for storing the boot code to be executed, said flash-based unit comprising a flash memory of a restricted type, being characterized in that code cannot be directly executed from said flash memory, and a volatile memory component for receiving a portion of the boot code to be executed, said portion of the boot code being for basic initialization of the system;

sending a busy signal to said processor;

transferring said portion of the boot code to said volatile memory component;

removing said busy signal; and

executing said portion of the boot code by said processor to boot the system.

22. A method for booting a system, the system featuring a processor for executing boot code, the method comprising:

providing a flash-based unit in the system for storing the boot code to be executed, said flash-based unit comprising a flash memory of a restricted type, being characterized in that code cannot be directly executed from said flash memory, and a volatile memory component for receiving a portion of the boot code to be executed;

transferring a first portion of the boot code to said volatile memory component, said first portion of the boot code being for basic initialization of the system and containing a command for copying a second portion of the code; and

executing said first portion of the boot code by said processor to boot the system.

23. The method of claim 22, further comprising the step of:

transferring a second portion of the code to said volatile memory component for booting the system.

24. A flash-based unit for providing boot code to be executed by an external processor, comprising:

(a) a flash memory for storing the boot code to be executed, said flash memory being of a type such that the boot code cannot be executed in place from said flash memory; and

(b) a volatile memory component for receiving at least a portion of the boot code to be executed, such that at least said portion of the boot code is executed by the external processor from said volatile memory component, said at least portion of the boot code being only sufficient for basic initialization of a system that includes the external processor, said volatile memory component being only large enough to store said at least portion of the boot code.

25. A system for executing boot code from a restricted non-volatile memory, the restricted non-volatile memory being characterized in that code cannot be directly executed from the restricted non-volatile memory, the system comprising:

(a) a CPU for executing the boot code; and

(b) a volatile memory component in direct communication with the restricted non-volatile memory for holding at least a portion of the boot code to be executed, said at least portion of the boot code being transferred from the restricted non-volatile memory, such that said CPU executes said at least portion of the boot code from said volatile memory component, said at least portion of the boot code being only sufficient for basic initialization of the system, said volatile memory component being only large enough to store said at least portion of the boot code.

26. A system for executing boot code, comprising:

(a) a flash-based unit for storing the boot code to be executed, said flash-based unit comprising a flash memory of a restricted type, being characterized in that the boot code cannot be directly executed from said flash memory, and a volatile memory component for receiving a portion of the boot code to be executed, said portion of the boot code being only sufficient for basic initialization of the system, said volatile memory component being only large enough to store said at least portion of the boot code; and

(b) a processor for executing the boot code, said processor receiving at least said portion of the boot code from said volatile memory component;

wherein an additional memory component is not required for executing the boot code by said processor.

27. A flash-based unit for providing boot code to be executed by an external processor, consisting essentially of:

(a) a flash memory for storing the boot code to be executed, said flash memory being of a type such that the boot code cannot be executed in place from said flash memory, and

(b) a volatile memory component for receiving at least a portion of the boot code to be executed, such that at least said portion of the boot code is executed by the external processor from said volatile memory component, said at least portion of the boot code being only sufficient for basic initialization of a system that includes the external processor, said volatile memory component being only large enough to store said at least portion of the boot code.

28. A flash-based unit for providing boot code to be executed by an external processor, comprising:

(a) a flash memory for storing the boot code to be executed, said flash memory being of a type such that the external processor cannot read the boot code to be executed directly from said flash memory; and

(b) a volatile memory component for receiving at least a portion of the boot code to be executed, such that at least said portion of the boot code is executed by the external processor from said volatile memory component, said at least portion of the boot code being only sufficient for basic initialization of a system that includes the external processor, said volatile memory component being only large enough to store said at least portion of the boot code.

36. The method of claim 19, wherein said flash-based unit is separate from the processor.